A WAVELET-OPTIMIZED, VERY HIGH ORDER ADAPTIVE GRID *and* ORDER NUMERICAL METHOD

Leland Jameson [1]
Mitsubishi Heavy Industries, Ltd.
Advanced Technology Research Center
8-1, Sachiura, 1-Chome, Kanazawa-ku,
Yokohama, 236, Japan
email: lmj@atrc.mhi.co.jp

ABSTRACT

Differencing operators of arbitrarily high order can be constructed by interpolating a polynomial through a set of data followed by differentiation of this polynomial and finally evaluation of the polynomial at the point where a derivative approximation is desired. Furthermore, the interpolating polynomial can be constructed from algebraic, trigonometric, or, perhaps exponential polynomials. This paper begins with a comparison of such differencing operator construction. Next, the issue of proper grids for high order polynomials is addressed. Finally, an adaptive numerical method is introduced which adapts the numerical grid and the order of the differencing operator depending on the data. The numerical grid adaptation is performed on a Chebyshev grid. That is, at each level of refinement the grid is a Chebyshev grid and this grid is refined locally based on wavelet analysis.

Key Words: Wavelet, Spectral Methods, Finite Difference, Adaptive Methods, Very High Order Methods.

AMS(MOS): subject classification: 63N30, 65N13.

Abbreviated title: Very High Order Adaptive Method

[1]This research was supported in part by the National Aeronautics and Space Administration under NASA Contract No. NAS1-19480 while the author was in residence at the Institute for Computer Applications in Science and Engineering (ICASE), NASA Langley Research Center, Hampton, VA 23681-0001.

Contents

List of Figures

List of Tables

1 Introduction

One can argue that high order numerical methods are appropriate for problems in 1) the direct numerical simulation of turbulence, see [17], 2) flows with shocks and nonlinear physics, see [8] and 3) flows with smooth propagating structures such as those encountered in aeroacoustics. Assertion number 3) is based on convergence properties of the hp-refinement method in finite elements, see [20], [1], [11], in which convergence is very fast for high order polynomials as long as the function at hand is smooth. In addition, high order methods are more efficient for long time integration of unsteady flow problems, see [18].

This paper introduces a numerical method which combines very high order differencing with a wavelet-based grid and order selection mechanism. Here very high order differencing will be schemes of order greater than or equal to 8, i.e., perhaps 16, 20, or maybe order 32. Such high orders of accuracy can produce solutions which are very close to those produced by spectral methods. See [3] for a spectral method on arbitrary grids.

The numerical method introduced here is named the Wavelet-Optimized Finite Difference method 2, or WOFD2. WOFD2 is an extension of WOFD, see [15], [9], in which wavelets are used for grid refinement, see [16], for finite difference schemes. The new method extends this scheme to very high orders and also adapts the order of accuracy depending on the data.

Let us begin by studying various manners in which high order difference operators can be constructed.

2 Generating Difference Equations

Given a vector of N numbers $\vec{f}$ how can we get an approximate value of the derivative $\vec{f}'$ at the $i-th$ point and how good will this approximate value be. Generally speaking, the more elements around the $i-th$ point of $\vec{f}$ that are used to approximate $\vec{f}'$ the better the approximation will be. Common finite difference formulas are found by fitting a algebraic polynomial of degree q locally around the $i-th$ point of a vector $\vec{f}$ of evenly-spaced elements to obtain difference approximations of accuracy $q-1$. This section will generalize this concept to find the difference equations of arbitrary accuracy on arbitrary grids using algebraic, trigonometric, cosine and exponential polynomials. As

special cases, one can obtain all the usual finite difference formulas as well as the Fourier collocation and Chebyshev collocation spectral differential matrices.

Two methods of generating the differencing coefficients will be introduced. The first method explains how to set up a system of equations which will have as a solution the differencing coefficients. The second method is the derivation of differencing coefficients by interpolation. It is this second method which is used throughout the paper for the actual generation of difference equations.

2.1 Setting up a Linear System

The problem is to find a set of coefficients $\{r_k\}$ which combines the raw data in a vector $\vec{f}$ to provide an approximation to a derivative:

$$f'(x_j) = \sum_{k=left}^{k=right} r_k f(x_k). \tag{1}$$

If we require the above equation to be exact for polynomials, algebraic,trigonometric, cosine or exponential, then a linear system of equations can be solved to find an appropriate set of differencing coefficients $\{r_k\}$. Let $b(x)$ denote a fundamental basis element from which a basis can generated by taking powers of $b(x)$: $b(x) = x$, $b(x) = e^{ix}$, $b(x) = \cos(x)$, or $b(x) = e^x$. That is, we require that the derivative be exact up to a given order N on the numerical grid. The system of equations to be solved for a centered differentiation stencil is as follows:

$$n(b(x_j))^{n-1} b'(x_j) = \sum_{k=-L}^{L} r_k (b(x_{j+k}))^n, \tag{2}$$

¿From this equation one can generate a system of equations with N requirements, that N functions be differentiated exactly, and N degrees of freedom, the N differencing coefficients r_k. If one is near a boundary, then the stenciled is biased. Since this type of system is well-known for algebraic polynomials, an example for the less well-known trigonometric polynomials will be given.

Consider a trigonometric polynomial on a 3 point centered stencil. The first equation simply requires that the derivative of a constant be zero:

$$0 = r_{-1} + r_0 + r_1. \tag{3}$$

Note that is the same equation as for algebraic polynomials since $(x)^0 = (e^{ix})^0$. The next two equations come from requiring that the $n = 1$ mode is differentiated exactly:

$$ie^{ix_0} = r_{-1}e^{ix_{-1}} + r_0 e^{ix_0} + r_1 e^{ix_1}. \tag{4}$$

One now obtains the two equations from equating the real and imaginary parts. These three equations can be solved for the three coefficients r_{-1}, r_0, r_1.

Similarly, one can find the coefficients for higher order schemes by requiring that more modes be differentiated exactly. Note that no restrictions were placed on the grid. Differencing formulas can be found on arbitrary grids as easily as they can be found on uniform grids. Also, note that the Fourier spectral differentiation matrix can be found from the above procedure by requiring that the grid be uniform and that the differencing formulas have maximum accuracy on a given grid. That is, if one is working on a grid of size 33 then require that the first 16 modes and the zero-th mode are differentiated exactly.

2.2 Interpolation

A second approach, and the one used in this paper, is to generate differencing coefficients by first interpolating a polynomial through a set of data, followed by differentiation of this polynomial and evaluated at a grid point.

The main reason that differentiation was studied with a variety of types of differentiation operators was to find out if there was any advantage to using, say, trigonometric polynomials to differentiate as opposed to algebraic polynomials when the function to be differentiated was for example a Gaussian pulse. It seemed like an appropriate study to undertake given the current research activity in the area or aeroacoustics where one is often confronted with the need to computationally propagate some type of wave motion. The thought was that perhaps trigonometric polynomials might have some advantage at propagating wave motion over the more common algebraic polynomials. One of the conclusions of this section is that there is no advantage and that one should simply use algebraic polynomials for the generation of differencing equations. In fact, the only important issues involved with obtaining approximate derivatives is the order of the finite difference operator and the density of the numerical grid.

The most important reference for this section is [7]. The following four subsections will cite the interpolation formulas for the four types of interpolation, and hence differentiation, considered in this section.

2.2.1 Algebraic Polynomials

Interpolation with algebraic polynomials is probably the most common form of interpolation, and it is from this type of interpolation that common uniform grid finite difference methods can be found. Using the following formula one can find the finite difference coefficients for an arbitrary grid and of arbitrary order. One simply fits the polynomial to the data, followed by differentiation of the polynomial, and finally one evaluates the polynomial at the point of interest. The well-known Lagrange interpolation formula for algebraic interpolation is,

$$A_j(x) = \prod_{k=0,k\neq j}^{n} (x - x_k) / \prod_{k=0,k\neq j}^{n} (x_j - x_k). \tag{5}$$

$A_j(x_k) = \delta_{jk}$ For given values $w_0, w_1, ..., w_n$, the polynomial

$$p_n(x) = \sum_{k=0}^{n} w_k A_k(x). \tag{6}$$

in P_n and takes on these values at the points x_i:

$$p_n(x_k) = w_k, \tag{7}$$

for $k = 0, 1, ..., n$.

2.2.2 Trigonometric Polynomials

As seen from the previous section, one can also generate difference operators by using trigonometric functions as the fundamental interpolation elements. The following is the appropriate Lagrange-type interpolation formula, see [7]:

For $-\pi \leq x_0 < x_1 < ... < x_{2n} < \pi$ then

$$T_j(x) = \prod_{k=0,k\neq j}^{2n} \sin\frac{1}{2}(x - x_k) / \prod_{k=0,k\neq j}^{2n} \sin\frac{1}{2}(x_j - x_k). \tag{8}$$

The function,

$$T(x) = \sum_{k=0}^{2n} w_k T_k(x) \tag{9}$$

is the unique solution of the interpolation problem,

$$T(x_k) = w_k, \tag{10}$$

for $k = 0, 1, ..., 2n$. Again, one can derive finite difference coefficients by interpolating to a function, followed by differentiation of the interpolation polynomial and evaluation at the point of interest. The following section will prove that such difference equations obey order properties just as the usual difference equations derived from algebraic polynomials do.

2.2.3 Cosine Polynomials

The comparable Lagrange-style interpolation formula for cosine polynomials is the following, see [7].

Given $n + 1$ distinct points $0 \leq x_0 < x_1 < ... < x_n < \pi$. Set

$$C_j(x) = \prod_{k=0, k\neq j}^{n} (\cos x - \cos x_k) / \prod_{k=0, k\neq j}^{n} (\cos x_j - \cos x_k). \tag{11}$$

Then C_j is a cosine polynomial of order $\leq n$, $C_j(x) = \sum_{k=0}^{n} a_k \cos(kx)$, for which $C_j(x_k) = \delta_{jk}$. Given $n+1$ distinct values $w_0, w_1, ..., w_n$ there is a unique cosine polynomial of order $\leq n$, $C(x)$, for which $C(x_k) = w_k$, $k = 0, 1, ..., n$. It is

$$C(x) = \sum_{k=0}^{n} w_k C_k(x). \tag{12}$$

Note that $\frac{d}{dx} C(x)|_{0,\pi} = 0$ since $\frac{d}{dx} C_k(x)|_{0,\pi} = 0$ for all k. For this reason, difference operators based on cosine polynomials will not be considered in general, but will be compared in a later section to Chebyshev spectral methods. As above, the differencing coefficients are found by first fitting the trigonometric polynomial to the data, followed by differentiation of the polynomial and finally evaluation at the point of interest.

2.2.4 Exponential Polynomials

The final polynomial to be tested is the exponential polynomial,

$$E_j(x) = \prod_{k=0,k\neq j}^{n} (e^x - e^{x_k}) / \prod_{k=0,k\neq j}^{n} (e^{x_j} - e^{x_k}), \tag{13}$$

where the interpolation polynomial is,

$$E(x) = \sum_{k=0}^{n} w_k E_k(x), \tag{14}$$

and

$$E(x_k) = w_k. \tag{15}$$

2.3 Truncation Error and Differentiation Accuracy

The purpose of this section is to illustrate algebraically that one obtains differentiation order-of-accuracy properties for all four types of differentiation operators which are similar to the standard order-of-accuracy properties obtained with the usual algebraic interpolation. In short, if one interpolates an N order polynomial then one obtains a reduction in differentiation error of $(\frac{1}{2})^N$ when the density of the grid is doubled. This order of accuracy is obtained regardless of the type of polynomials which are used.

Recall that the remainder for algebraic polynomial interpolation is, see [7],

$$f(x) - p_n(x) = \frac{(x - x_0)(x - x_1)...(x - x_n)}{(n+1)!} f^{(n+1)}(\xi), \tag{16}$$

where ξ lies between the smallest and the largest x_i. The following section will show that a similar expression can be obtained for any polynomial constructed from powers of a given function.

2.3.1 Truncation Error for Interpolation by Powers

There are some subtle issues concerning a general proof of truncation error and accuracy for interpolation by a polynomial constructed from the powers of a general function $g(x)$, see [4]. The following demonstration will illustrate the essentual algebraic steps that one follows to obtain accurcy while avoiding

the subtle issues. In short, let the polynomial element $g(x)$ and the function to be approximated $f(x)$ be "well-behaved".

Let

$$p(x) = \sum_{k=0}^{n} a_k (g(x))^k$$

be the polynomial which interpolates $f(x)$ at $x_0, x_1, ..., x_n$, $p(x_i) = f(x_i)$, then,

$$f(x) - p(x) = \frac{p^{(n+1)}(\xi) - f^{(n+1)}(\xi)}{\phi^{(n+1)}(\xi)} \phi(x), \tag{17}$$

where

$$\phi(x) = (g(x) - g(x_0))(g(x) - g(x_1))...(g(x) - g(x_n)), \tag{18}$$

and where ξ lies between the smallest and the largest x_i.

Demonstration of Truncation Error: Note that much of this demonstration is the same as that which can be found in a standard numerical analysis text for the remainder term in algebraic interpolation, see [5].

Define $H(z)$ such that

$$H(z) = f(z) - p(z) - R(x)\phi(z), \tag{19}$$

where $R(x)$ is defined such that $H(x) = 0$. Note that $H(x_i) = 0$, for $i = 0, ..., n$ since $p(x_i) = f(x_i)$ and $\phi(x_i) = 0$. ¿From Rolle's theorem it follows that there exists a point ξ in the interval defined by the smallest and largest x_i's such that $H^{(n+1)}(\xi) = 0$. This implies,

$$R(x) = \frac{f^{(n+1)}(\xi) - p^{(n+1)}(\xi)}{\phi^{(n+1)}(\xi)}. \tag{20}$$

Now put this back into the expression for $H(z)$ and set $z = x$ to get,

$$f(x) - p(x) = \frac{f^{(n+1)}(\xi) - p^{(n+1)}(\xi)}{\phi^{(n+1)}(\xi)} \phi(x). \tag{21}$$

This is the desired expression. //.

Note that in the above demonstration that if the polynomial is algebraic that $p^{(n+1)}(z) = 0$ and $\phi^{(n+1)}(z) = (n+1)!$, but for a general $g(x)$ these two functions are just a measure of the smoothness of the basic interpolation element, $g(x)$. $f^{(n+1)}(\xi)$ still remains a measure of the smoothness of the function one is interpolating to, and $\phi(x)$ is a function dependent on the grid distribution.

2.3.2 Differentiation Accuracy

The primary interest here is to understand the behavior of the derivative operators derived from the various types of interpolation outlined above as the grid is refined. That is,

$$f'(x) - p'(x) = Q(\xi)\phi'(x), \tag{22}$$

where $Q(\xi) = \frac{f^{(n+1)}(\xi) - p^{(n+1)}(\xi)}{\phi^{(n+1)}(\xi)}$, and $\phi'(x)$ will dictate the behavior as the grid is refined. It will be shown that the behavior of $\phi'(x)$ is essentially independent of the basic interpolation element $g(x)$ and depends only on the order of the interpolation.

Demonstration of Accuracy:
Let h denote the smallest spacing in the numerical grid, then

$$\phi'(x_0) = Ch^n + h.o.t. \tag{23}$$

where n is the highest power in the interpolation polynomial, and the point x_0 is an arbitrary grid point inside the interpolation stencil.

Demonstration: First of all,

$$\phi'(x_0) = g'(x_0)(g(x_0) - g(x_1)(g(x_0) - g(x_2))...(g(x_x) - g(x_n)). \tag{24}$$

Expand about zero the function $g(x)$,

$$g(x) = g(0) + g'(0)x + g''(0)x^2/2 + ... \tag{25}$$

and examine the difference $g(x_0) - g(x_1)$

$$g(x_0) - g(x_1) = g'(0)(x_0 - x_1) + g''(0)/2(x_0^2 - x_1^2) + g'''(0)/6(x_0^3 - x_1^3) + ... \tag{26}$$

Without loss of generality let one of the points be zero, say $x_0 = 0$, to get,

$$g(0) - g(x_1) = g'(0)(-x_1) + g''(0)/2(-x_1^2) + g'''(0)/6(-x_1^3) + ..., \quad (27)$$

or,

$$g(0) - g(x_1) = -x_1(\sum_{m=1}^{\infty} \frac{g^{(m)}(0)}{m!} x_1^{m-1}), \quad (28)$$

and one can see that the first term in the difference is linear. If x_0 is not zero then one obtains the factorization,

$$x^q - y^q = (x - y)(\sum_{i=0}^{q-1} x^i y^{q-1-i}), \quad (29)$$

and hence,

$$g(x_0) - g(x_1) = (x_0 - x_1)(\sum_{m=1}^{\infty} \frac{g^{(m)}(0)}{m!} \sum_{k=0}^{m-1} x_0^k x_1^{m-1-k}). \quad (30)$$

It should be clear that the first term in the difference $g(x_0) - g(x_1)$ is the linear term and that doubling the grid such that between every two points another point is placed is, therefore, halves the distance to a first order approximation. For each of the differences $x_i - x_j$ there exists a constant $c_{i,j}$ such that the difference,

$$x_i - x_j = c_{i,j} h \quad (31)$$

can be expressed in terms of the smallest grid spacing h. Let $C = \prod_{j=1}^{n} c_{0,j}$ then it is, therefore, clear that

$$\phi'(x_0) = Ch^n + h.o.t. \quad (32)$$

//.

Consider the following special cases which includes all the polynomial types discussed above:

- $g(x) = x$, $g^{(m)}(0) = 0$, for $m \neq 1$
- $g(x) = e^x$, $g^{(m)}(0) = 1$, $\forall m$

- $g(x) = e^{ix}$, $g^{(m)}(0) = i^m$
- $g(x) = \cos(x)$, $g^{(m)}(0) = 0$, for m odd and $g^{(m)}(0) = (-1)^{m/2}$, for m even

For a simple illustration, consider the following two examples of algebraic and exponential interpolation.

Algebraic Interpolation

Consider the simple case of interpolating an algebraic quadratic polynomial $p_2(x)$ to a function $f(x)$ at the grid points $x_0 < x_1 < x_2$: $p_2(x_i) = f(x_i)$, $i = 0, 1, 2$. The remainder term for some ξ, $x_0 \leq \xi \leq x_2$, is

$$f(x) - p_2(x) = (x - x_0)(x - x_1)(x - x_2)\frac{1}{3!}f^{(3)}(\xi). \tag{33}$$

Now, differentiate and evaluate at $x = x_1$ to get,

$$f'(x_1) - p_2'(x_1) = (x_1 - x_0)(x_1 - x_2)\frac{1}{3!}f^{(3)}(\xi) = Ch^2 f^{(3)}(\xi), \tag{34}$$

where $h = x_1 - x_0 = x_2 - x_1$. If the grid is evenly-spaced then the differences $(x_i - x_j)$ are some integer multiple of the smallest difference which one can denote by h. If one doubles the number of grid points then each of the distances $(x_i - x_j)$ becomes half as large and the accuracy for this quadratic example will be 2.

The General Statement for Algebraic Interpolation

In general, one can expect that algebraic polynomial interpolation with a polynomial of order n, $p_n(x)$, will produce a differencing operator of order also of order n. This can be seen from the portion of the truncation error which depends on the grid distribution:

$$\prod_{i=0}^{n}(x - x_i). \tag{35}$$

This product contains $n + 1$ terms. After differentiation and evaluation at a point x_k the product will contain n terms,

$$\prod_{i=0}^{n-1}(x_k - x_i). \tag{36}$$

When the grid density is doubled by adding a point between every two points, then each distance $(x_k - x_i)$ will decrease by a factor of 2. The product of these factors will be a multiple of $(\frac{1}{2})^n$ and hence the accuracy of n is achieved.

Exponential Interpolation

If, on the other hand, $p_2(x)$ is now an exponential polynomial then after differentiation and evaluation one gets,

$$f'(x_1) - p_2'(x_1) = e^{x_1}(e^{x_1} - e^{x_0})(e^{x_1} - e^{x_2})\frac{1}{3!}f^{(3)}(\xi) \tag{37}$$

Without loss of generality, let $x_1 = 0$. Then the product,

$$f'(0) - p_2'(0) = (e^{x_1} - e^{x_0})(e^{x_1} - e^{x_2}) \tag{38}$$

becomes

$$f'(0) - p_2'(0) = (x_0 + x_0^2/2 + ...)(x_2 + x_2^2/2 + ...), \tag{39}$$

and one can see that if the distance to zero is halved and hence x_0 and x_2 are divided by 2 that the leading order terms in each of the above parenthesis dictates that the error will be reduced by 4 to a first order approximation.

Therefore, one can expect the accuracy to behave as it does for algebraic polynomials. That is, doubling the grid points will decrease the error by $(1/2)^2$ to first order.

The General Statement for Exponential Interpolation

In general, one can expect that exponential polynomial interpolation with a polynomial of order n, $p_n(x)$, will produce a differencing operator of order also of order n which is the same result as for algebraic interpolation. As can be seen from the above example, differentiation and evaluation at a point x_k will produce a leading order term which is a product of n terms,

$$\prod_{i=0}^{n-1}(x_k - x_i), \tag{40}$$

and accuracy of order n is achieved.

Grid Pts	Alg Err	Err Ratio	Trig Err	Err Ratio	Exp Err	Err Ratio
16	$7.72 * 10^{-4}$		$8.23 * 10^{-4}$		$4.10 * 10^{-3}$	
32	$4.06 * 10^{-6}$	$2^{7.6}$	$3.66 * 10^{-6}$	$2^{7.8}$	$1.25 * 10^{-5}$	$2^{8.4}$
64	$8.02 * 10^{-9}$	$2^{9.0}$	$7.59 * 10^{-9}$	$2^{8.9}$	$1.95 * 10^{-8}$	$2^{9.3}$
128	$9.54 * 10^{-12}$	$2^{9.7}$	$8.75 * 10^{-12}$	$2^{9.8}$	$2.17 * 10^{-11}$	$2^{9.8}$
256	$9.80 * 10^{-15}$	$2^{9.9}$	$9.04 * 10^{-15}$	$2^{9.9}$	$2.20 * 10^{-14}$	$2^{9.9}$
512	$9.70 * 10^{-18}$	$2^{10.0}$	$8.95 * 10^{-18}$	$2^{10.0}$	$2.17 * 10^{-17}$	$2^{10.0}$

Table 1: A Comparison of Errors for Various Types of Differentiation Operators

2.4 A Numerical Check of Accuracy

This subsection will verify that the differentiation operators which are generated from algebraic, trigonometric, and exponential polynomials all exhibit the same order property, which depends only on the order of the polynomial interpolation. The difference operators will be tested on the function,

$$f(x) = \frac{1}{2 + \cos(2x)} \tag{41}$$

defined on $[0, \pi]$. $f(x)$ is chosen because it is periodic but not exactly a trigonometric or algebraic polynomial. Table (1) illustrates the order property All of the errors are L_2.

A general question arises, is there any advantage to using, say, a trigonometric polynomial for the generation of difference equations over, say, the usual algebraic polynomial? From this study the answer appears to be no. The essence depends on the ability of the interpolating polynomial to locally approximate the function at hand. If the function at hand is not exactly a trigonometric or algebraic polynomial, as is likely, then there is no advantage for either approach. Such an issue is important when one is considering the propagation of, say, a pulse in the application of aeroacoustics. A pulse will locally be neither a algebraic or trigonometric polynomial. In short, a wave,

or pulse, can not be propagated accurately if it can not first be differentiated accurately, and it can not be differentiated accurately if it can not first be approximated accurately.

3 High Order Methods

Spectral collocation methods are often given the probable misnomer of "infinitely accurate". In a manner consistent with finite difference methods, the accuracy of spectral collocation methods will be assigned the accuracy of $N-1$ when applied on a grid of N points.

This section will begin by connecting spectral collocation methods to finite difference methods. That is, spectral methods will be viewed from the point of view of the maximum finite difference method on a given grid. Following the comments on this connection, a case will be made for applying very high order algebraically generated finite difference operators on Cheybshev grids or, equivalently, applying very high order cosine polynomial generated finite difference operators on a uniform grid. This second process of using cosine polynomials will require a mapping of the independent variable from, say, x to $\cos(x)$, but is exactly equal to applying algebraic polynomials on Chebyshev grids.

3.1 Spectral Collocation = Maximum Order Finite Difference

On a numerical grid of N points one can fit a polynomial with N degrees-of-freedom through all of the data. If this polynomial is algebraic and if the grid distribution is Chebyshev, $x_i = \cos(\frac{i\pi}{N})$, then one can build the Chebyshev collocation differentiation matrix. On the other hand, if the polynomial is trigonometric and if the grid is uniformly distributed then one can build the Fourier spectral collocation differentiation matrix. One can, therefore, define in the physical space a spectral method to be a method which uses the maximum size polynomial for approximation and differentiation for a given grid size.

Another way to see this is, suppose one has a numerical grid of 16 points and a 4-th order difference operator on a 5 point stencil. Now reduce the number of grid points to 8. The difference operator is still 4-th order. Now

Grid Pts	sin Freq	L_2 Error
9	1.0	1.6710^{-28}
9	1.5	8.7110^{-1}
9	2.0	5.6810^{-29}
9	2.5	1.6010^{0}
9	3.0	2.2410^{-28}
9	3.5	2.7910^{0}
9	4.0	8.010^{-28}
9	4.5	2.9110^{0}
9	5.0	3.0510^{0}

Table 2: Fourier Spectral Collocation Applied to sin's

reduce the number of grid points to 5. The difference operator is still 4-th order accurate. This is spectral accuracy. That is, spectral accuracy of collocation methods on finite grids is $N-1$ where N is the number of grid points.

3.1.1 A Numerical Check

Let us consider the above statements numerically. A Fourier collocation spectral method on a grid of 9 points is a differencing mechanism with exactly 9 degrees-of-freedom, and hence, is designed to differentiate exactly 9 functions exactly: 1, $\cos(kx)$, and $\sin(kx)$, for $k = 1, 2, 3, 4$. Table (2) is meant to illustrate two points: i) the maximum frequency which is differentiated exactly is 4.0, and ii) the poor performance on non-integer frequencies.

Likewise, a Chebyshev collocation spectral method on a grid of 9 points is designed to differentiate 9 functions exactly: x^k for $k = 0, ..., 8$. Table (3) is designed to illustrate the same two points at Table (2). The table begins with the function x^5.

Note that if the function being differentiated is not exactly an integer frequency, e^{ikx} or x^k, then, say for Chebyshev, differentiating $x^{5.5}$ is no better or worse than the result for differentiating $\sin(5.5x)$. The point that is trying to be made is that the differentiation accuracy of spectral collocation

Grid Pts	Poly Order	L_2 Error
9	x^5	2.2910^{-25}
9	$x^{5.5}$	9.2610^{-4}
9	x^6	6.4910^{-25}
9	$x^{6.5}$	2.7710^{-3}
9	x^7	2.3910^{-24}
9	$x^{7.5}$	1.7910^{-2}
9	x^8	6.9610^{-24}
9	$x^{8.5}$	4.2910^{-1}
9	x^9	2.8310^{0}

Table 3: Chebyshev Spectral Collocation Applied to Polynomials

methods on a finite grid of size N is accurate with the accuracy of $N-1$, and one can not expect that a wave-like pulse will be transmitted better with a Fourier spectral method than with a Chebyshev spectral. The only important issue is the dimension of the space and the boundary conditions: use Chebyshev for non-periodic boundary conditions and Fourier for period boundary conditions.

3.2 Very High Order Finite Differencing

Now suppose that build a series of algebraically generated finite difference operators of increasing accuracy and test these difference operators on the function $\sin(2x)$. The grid size will be fixed at 33 points. The first two lines of Table (4) illustrate the effect of the Runge phenomenon, see [5]. The change in the error from periodic boundary conditions on a uniform grid to non-periodic boundary conditions on a uniform grid is from 10^{-27} to 10^{-21}. In addition, note that applying an algebraic polynomial with periodic boundary conditions yields a result comparable to applying a trigonometric, i.e. Fourier spectral collocation, polynomial with periodic boundary conditions. Furthermore, one can observe the Runge phenomenon with trigonometric polynomials just as one observes it with algebraic polynomials when the boundary conditions are not periodic. Note that 128 bit arithmetic is being

Grid Pts	Order of Acc	L_2 Error	Error Ratio	Periodic BC's	Grid even or Cheby
33	32	1.5210^{-27}		yes	even
33	32	5.0710^{-21}		no	even
33	18	7.4910^{-17}		no	Cheby
33	20	1.1710^{-18}	64.0	no	Cheby
33	22	1.7310^{-20}	67.6	no	Cheby
33	24	2.4310^{-22}	71.2	no	Cheby
33	26	3.4910^{-24}	69.6	no	Cheby
33	28	5.7510^{-26}	60.7	no	Cheby
33	30	1.3010^{-27}	44.2	no	Cheby
33	32	8.8910^{-28}	1.46	no	Cheby

Table 4: Finite Difference Accuracy Approaching Spectral Accuracy

used. ¿From line 3 to the bottom of the table the order of accuracy is increased from 18 to the maximum, i.e., spectral, accuracy of 32 is obtained. When one tests the accuracy of a finite difference operator one doubles the grid and sees the error decrease as $(\frac{1}{2})^n$ where n is the accuracy of the scheme. This comes from the truncation error which will produce a factor of the form $(\Delta x)^n$. In Table (4), it is the number n which is being increased while Δx remains constant.

Compare Table (4) to Table (5) in which a Chebyshev collocation method on an increasing grid size is tested on $\sin(2x)$. Note that in the following table that both Δx is decreasing and n is increasing in the expression $(\Delta x)^n$ as one proceeds down the table. The final line of Table (5) is the Chebyshev method on a grid of 33 points which produces a result comparable the result in Table (4) on a grid of 33 points. The numbers are not exactly the same because, first of all, all calculations are near machine accuracy, and, second, the differencing coefficients are calculated in different ways.

Grid Pts	Alg Err	Err Ratio
9	3.0010^{-3}	
11	8.6010^{-5}	34.9
13	1.6510^{-6}	52.1
15	2.2910^{-8}	72.1
17	2.3810^{-10}	96.2
19	1.9410^{-12}	122.7
21	1.2710^{-14}	152.8
23	6.8510^{-17}	185.4
25	3.0810^{-19}	222.4
27	1.1710^{-21}	263.2
29	3.8510^{-24}	303.9
31	1.0910^{-26}	353.2
33	2.5410^{-27}	4.3

Table 5: Chebyshev Spectral Collocation of Increasing Order

3.3 Chebyshev Spectral Methods and Cosine Polynomials

This subsection will review Chebyshev spectral methods and the equivalency with cosine polynomials. Chebyshev approximation can be seen as approximation by algebraic polynomials,

$$T_0(x) = 1, \tag{42}$$

$$T_1(x) = x,$$

$$T_2(x) = 2x^2 - 1,$$

$$T_3(x) = 4x^3 - 3x,$$

or as approximation by a cosine series, see [7],

$$T_n(x) = \cos(n \arccos x) = \cos(n\theta) = \sum_{q=0}^{n} a_q (\cos\theta)^q = \sum_{q=0}^{n} a_q x^q, \tag{43}$$

for some set $\{a_q\}$ and where $x = \cos(\theta)$. If one now chooses a numerical grid defined as $x_j = \cos(\frac{\pi j}{N})$, $j = 0, ..., N$ then one obtains $T_n(x_j) = \cos(\frac{\pi jn}{N})$ and the pseudospectral Chebyshev method, see [10], [21], and [2].

A Chebyshev spectral method involves approximating a function, $f(x)$, by interpolating an algebraic polynomial to point values $f(x_i)$ where the grid points are given by the 'Chebyshev' grid points $x_i = \cos(\theta_i)$. An equivalent process is to interpolate a cosine polynomial to the evenly-spaced point values of the angle θ_i and to consider $f(x)$ to be evaluated on the uniform grid of the angle variable θ_i, $f(x_i)$ becomes $f(\theta_i)$. That is, see [10], if the Chebyshev series for $f(x)$ is

$$Pf(x) = g(x) = \sum_{k=0}^{\infty} a_k T_k(x), \tag{44}$$

then the expansion coefficients $\{a_k\}$ can be found in two equivalent ways,

$$a_k = \frac{2}{\pi c_k} \int_{-1}^{1} f(x) T_k(x) (1 - x^2)^{-1/2} dx = \frac{2}{\pi c_k} \int_0^{\pi} f(\cos\theta) \cos k\theta d\theta \tag{45}$$

By this transformation of the independent variable one can perform Chebyshev spectral methods on a uniform grid or one can build arbitrarily high difference operators on a uniform grid which have stability characteristics equivalent to the usual Chebyshev spectral method.

3.4 High-Order Differencing on Chebyshev Grids

Chebyshev spectral methods work very well for non-periodic problems precisely because the truncation error for a Chebyshev polynomial is equil-ripple.

As shown above, the truncation error for polynomial approximation, $p_n(x)$, of a function $f(x)$ is

$$f(x) - p_n(x) = \frac{(x - x_0)(x - x_1)...(x - x_n)}{(n + 1)!} f^{n+1}(\xi), \tag{46}$$

where ξ lies between the smallest and the largest x_i. If one wants to minimize the error due to the term

$$(x - x_0)(x - x_1)...(x - x_n)$$

then the $n+1$ sample points should be chosen as the zeros of Chebyshev polynomial $T_{n+1}(x)$, see [7]. This selection of grid points ensures that the error has the equal-ripple property which is characteristic of Chebyshev polynomials. A Chebyshev spectral method interpolates the highest order polynomial possible onto the $n + 1$ degrees-of-freedom defined by the point values of a function at the $n + 1$ zeros of $T_{n+1}(x)$. The question to be addressed now is what if the grid is defined as the zeros of $T_{n+1}(x)$ but the polynomial is of lower order. That is, n could be, say, 128 whereas the polynomial on this grid could be of order 16. It is well-known that high order polynomial interpolation on uniform grids is essentially an ill-posed problem.

4 Wavelet-based Grid and Order Selection

The previous section introduced the idea of building very high order algebraically-generated difference operators on Chebyshev grids as a way of obtaining very high accuracy which is almost spectral in nature. This section will explore the idea of performing wavelet-based grid refinement on these Chebyshev grids as a way to obtain the necessary Chebyshev grid distribution near a boundary while having the ability to refine the grid away from the boundary for proper physical-space function resolution.

4.1 A Short Review of Wavelets

To define Daubechies-based wavelets, see [6] for the original work and see [19] for an introduction to wavelet-based signal processing, consider the two functions $\phi(x)$, the scaling function, and $\psi(x)$, the wavelet. The scaling function is the solution of the dilation equation,

$$\phi(x) = \sqrt{2} \sum_{k=0}^{L-1} h_k \phi(2x - k), \tag{47}$$

where $\phi(x)$ is normalized $\int_{-\infty}^{\infty} \phi(x)dx = 1$, and the wavelet $\psi(x)$ is defined in terms of the scaling function,

$$\psi(x) = \sqrt{2} \sum_{k=0}^{L-1} g_k \phi(2x - k). \tag{48}$$

One builds an orthonormal basis from $\phi(x)$ and $\psi(x)$ by dilating and translating to get the following functions:

$$\phi_k^j(x) = 2^{-\frac{j}{2}} \phi(2^{-j}x - k), \tag{49}$$

and

$$\psi_k^j(x) = 2^{-\frac{j}{2}} \psi(2^{-j}x - k), \tag{50}$$

where $j, k \in Z$. j is the dilation parameter and k is the translation parameter. The coefficients $H = \{h_k\}_{k=0}^{L-1}$ and $G = \{g_k\}_{k=0}^{L-1}$ are related by $g_k = (-1)^k h_{L-k}$ for $k = 0, ..., L-1$. All wavelet properties are specified through the parameters H and G. If one's data is defined on a continuous domain such as $f(x)$ where $x \in R$ is a real number then one uses $\phi_k^j(x)$ and $\psi_k^j(x)$ to perform the wavelet analysis. If, on the other hand, one's data is defined on a discrete domain such as $f(i)$ where $i \in Z$ is an integer then the data is analyzed, or filtered, with the coefficients H and G. In either case, the scaling function $\phi(x)$ and its defining coefficients H detect localized low frequency information, i.e., they are low-pass filters (LPF), and the wavelet $\psi(x)$ and its defining coefficients G detect localized high frequency information, i.e., they are high-pass filters (HPF). Specifically, H and G are chosen so that dilations and translations of the wavelet, $\psi_k^j(x)$, form an orthonormal basis of $L^2(R)$ and so that $\psi(x)$ has M vanishing moments which determines the accuracy. In other words, $\psi_k^j(x)$ will satisfy

$$\delta_{kl}\delta_{jm} = \int_{-\infty}^{\infty} \psi_k^j(x)\psi_l^m(x)dx, \tag{51}$$

where δ_{kl} is the Kronecker delta function, and the accuracy is specified by requiring that $\psi(x) = \psi_0^0(x)$ satisfy

$$\int_{-\infty}^{\infty} \psi(x) x^m dx = 0, \tag{52}$$

for $m = 0, ..., M-1$. Under the conditions of the previous two equations, for any function $f(x) \in L^2(R)$ there exists a set $\{d_{jk}\}$ such that

$$f(x) = \sum_{j \in Z} \sum_{k \in Z} d_{jk} \psi_k^j(x), \tag{53}$$

where

$$d_{jk} = \int_{-\infty}^{\infty} f(x) \psi_k^j(x) dx. \tag{54}$$

The two sets of coefficients H and G are known as quadrature mirror filters. For Daubechies wavelets the number of coefficients in H and G, or the length of the filters H and G, denoted by L, is related to the number of vanishing moments M by $2M = L$. For example, the famous Haar wavelet is found by defining H as $h_0 = h_1 = 1$. For this filter, H, the solution to the dilation equation (47), $\phi(x)$, is the box function: $\phi(x) = 1$ for $x \in [0, 1]$ and $\phi(x) = 0$ otherwise. The Haar function is very useful as a learning tool, but because of its low order of approximation accuracy and lack of differentiability it is of limited use as a basis set. The coefficients H needed to define compactly supported wavelets with a higher degree of regularity can be found in [6]. As is expected, the regularity increases with the support of the wavelet. The usual notation to denote a Daubechies-based wavelet defined by coefficients H of length L is D_L.

It is usual to let the spaces spanned by $\phi_k^j(x)$ and $\psi_k^j(x)$ over the parameter k, with j fixed, be denoted by V_j and W_j respectively,

$$V_j = \underset{k \in Z}{span}\ \phi_k^j(x), \tag{55}$$

$$W_j = \underset{k \in Z}{span}\ \psi_k^j(x). \tag{56}$$

The spaces V_j and W_j are related by,

$$... \subset V_1 \subset V_0 \subset V_{-1} \subset ..., \tag{57}$$

and

$$V_j = V_{j+1} \bigoplus W_{j+1}, \tag{58}$$

where the notation $V_0 = V_1 \bigoplus W_1$ indicates that the vectors in V_1 are orthogonal to the vectors in W_1 and the space V_0 is simply decomposed into these two component subspaces.

The previously stated condition that the wavelets form an orthonormal basis of $L^2(R)$ can now be written as,

$$L^2(R) = \bigoplus_{j \in Z} W_j. \tag{59}$$

Two final properties of the spaces V_j are that,

$$\bigcap_{j \in Z} V_j = \{0\}, \tag{60}$$

and

$$\overline{\bigcup_{j \in Z} V_j} = L^2(R). \tag{61}$$

4.2 Grid Refinement on Uniform Grids

The idea of using wavelets to generate numerical grids began with the observation in [12] that the essence of an adaptive wavelet-Galerkin method is nothing more than a finite difference method with grid refinement. So, instead of letting the magnitude of wavelet coefficients choose which basis functions to use in a Galerkin approach, let the same coefficients choose which grid points to use and then think of the wavelet method in a collocation sense.

In other words, suppose a calculation begins with N evenly-spaced samples of a function $\vec{f}$ and that some quadrature method produces N scaling function coefficients on the finest scale denoted by V_0. If the spacing between adjacent values in the vector $\vec{f}$ is Δx then this is also the physical-space resolution of any calculation done in V_0. Now, decompose V_0 once to get $V_0 = V_1 \oplus W_1$. Similarly speaking, the physical space resolution of V_1 is $2\Delta x$ and the refinement from the $2\Delta x$ physical-space resolution to the Δx physical-space resolution is dictated by the wavelet coefficients in W_1. This is the reasoning which led to WOFD and to the following subroutine which is at the heart of WOFD. The remainder of the paper is concerned with giving the reader an idea of how the WOFD grid refinement software works.

4.3 Grid Refinement on Chebyshev Grids

Chebyshev grids are not evenly-spaced in physical space, but are evenly-spaced in angle. That is, a Chebyshev grid comes from $x_j = \cos(\theta_j)$, $j = 0, ..., N$, where the angle $\theta_j = \frac{\pi j}{N}$ is evenly-spaced. The above described refinement mechanism can now be applied to the uniform angle grid point values to define a new numerical grid. That is, all the above grid refinement machinery can be applied to Chebyshev grids where each subspace V_j will coincide with a uniform angle or usual Chebyshev grid and each refinement subspace W_j will coincide with additional points being added to the usual Chebyshev grid. It is well known that the Chebyshev grid is the best grid, in terms of minimal error, for algebraic polynomial interpolation. A refinement, W_1, on Chebyshev grid, V_1, to get $V_0 = V_1 \oplus W_1$ is disigned to begin with a grid which in one sense in perfect, the Chebyshev grid, and perturb from this grid.

5 A New Numerical Method: WOFD2

In [15] a numerical method was defined which was called the Wavelet-Optimized Finite Difference method or WOFD. WOFD used wavelets in their finite difference form. In essence, this meant that wavelets were used to choose a numerical grid and all computations were performed on this grid using arbitrary-grid finite difference operators.

WOFD2 is an extension of this idea. WOFD2 uses wavelets to choose not only a numerical grid but also the order of the difference operator used on this grid. In addition, WOFD2 uses very high order finite difference operators on the order of 8, 16 or maybe 32. Furthermore, the physical-space grids are no longer evenly-spaced at every resolution but are Chebyshev. That is, wavelet-based grid generation, see [16], requires that a grid be selected from a uniform finest grid. But, high order polynomials can be highly-oscillatory on uniform grids. Therefore, WOFD2 works with Chebyshev grids at each resolution level. Recall, that Chebyshev grids $x_i = \cos(\theta_i)$ are not uniform in the physical space variable x_i but are uniform in the angle variable θ_i. It is in this uniform angle variable θ_i that grid refinement is performed.

Using the grid selection mechanism in [16] to select order is a minor extension of the idea that wavelets are very good at finding regions of the domain

at which a large numerical error is likely to occur. Numerical error will determined by the truncation error of a polynomial which is locally interpolated to the data. The truncation error will be the product of intervals and a constant. Imagine the intervals are all a multiple of a smallest interval Δx then the key component in the truncation error will be $(\Delta x)^n$. This component can be decreased by either decreasing the size of Δx or by increasing the order of the scheme, i.e., increasing n. Or, one can decrease Δx and increase n simultaneously.

5.1 Comparison with hp-Refinement

In the finite element literature, see [20], [1], [11], the idea of refining the grid and increasing the polynomial order is known as hp-refinement. The theory from hp-refinement can certainly be applied to the new method WOFD2, even though WOFD2 works with polynomials only for generation of finite difference operators to be applied in the physical space. One of the most important results from hp-refinement theory from which WOFD2 can benefit is that when the function being differentiated is smooth then the rate of convergence is controlled by the polynomial degree. For the purpose of pulse propagation in aeroacoustics it is apparent that a high order differentiation, i.e., high order polynomial interpolation, will propagate the pulse more faithfully than grid refinement on the same pulse, assuming the pulse is smooth.

5.2 Numerical Experiments with WOFD2

This section will provide the results from numerous numerical experiments performed with WOFD2. For all the numerical experiments in this section of the paper a Gaussian pulse enters the domain from the right-hand side and travels to the left. The governing equation is the 1 dimensional hyperbolic wave equation,

$$U_t(x,t) = U_x(x,t), \;\; U(x,0) = e^{-c(x-\pi)^2}, \tag{62}$$

for some constant c. See Figure 1 for a plot of this initial condition. Note that the domain extends from 0 to π. The final time for all simulation is $\pi/2$. The simulation is stopped at this value because at $\pi/2$ the Chebyshev grid has a maximum spacing between grid points and hence a minimum resolution.

Grid Pts	Order of Acc	L_2 Error	L_∞ Error	Final Time
64	8	$3.82 * 10^{-3}$	$1.78 * 10^{-2}$	$\pi/2$
64	16	$2.95 * 10^{-4}$	$1.30 * 10^{-3}$	$\pi/2$
64	32	$1.74 * 10^{-5}$	$7.10 * 10^{-5}$	$\pi/2$
64	48	$2.74 * 10^{-6}$	$1.28 * 10^{-5}$	$\pi/2$

Table 6: WOFD2 of Accuracies 8, 16, 32, and 48

5.2.1 No Adaptation

First we consider the case of very high order finite differencing on a Chebyshev grid. The grid size is kept fixed at 64 points, and the order is increased from 8 to 48. The errors decrease in a nice and uniform manner. No unusual numerical oscillations occur.

5.2.2 Adapting Grid Only: Order 8 Spatial Differencing

In this subsection the order of the spatial differencing is kept fixed at order 8. No results are found for threshold values of 10^{-1} and 10^{-2}. This is because it seems to be a characteristic of adaptive methods that a very rough threshold value can degrade the performance of the method. It is better to start with threshold values less than or equal to 10^{-3}. The first row of the table is the worst possible performance where no refinement is done and grid is 64 points, and the last row of the table is best possible performance where the grid is 128 points. Note that the software is constructed to work with both periodic and non-periodic boundary conditions, so that when the boundary conditions are non-periodic the possible number of points becomes 2^N+1 which includes the right-hand boundary point. For periodic boundary conditions the number of grid points is 2^N since the right-hand boundary point is equal to the first point on the left-hand boundary.

Grid Pts	t_f Grid	Order of Acc	L_2 Error	L_∞ Error	Thresh	Final Time
128/64	65	8	$3.82 * 10^{-3}$	$1.78 * 10^{-2}$	100.0	$\pi/2$
128/64	77	8	$1.67 * 10^{-3}$	$8.66 * 10^{-3}$	10^{-3}	$\pi/2$
128/64	79	8	$1.63 * 10^{-4}$	$8.76 * 10^{-4}$	10^{-4}	$\pi/2$
128/64	82	8	$8.24 * 10^{-5}$	$3.24 * 10^{-4}$	10^{-5}	$\pi/2$
128/64	84	8	$8.07 * 10^{-5}$	$3.24 * 10^{-4}$	10^{-6}	$\pi/2$
128	129	8	$6.51 * 10^{-5}$	$3.24 * 10^{-4}$	0.0	$\pi/2$

Table 7: WOFD2 Adaptive Grid, but Accuracy fixed at 8

5.2.3 Adapting Grid Only: Order 16 Spatial Differencing

Much of what was said for the 8th order table above can be said here. The second row of the following table shows how the performance can be slightly degraded for relatively large threshold values. In this case the degradation occurs at the threshold value of 10^{-3}. This is a minor point. Generally speaking, just start with a smaller threshold value.

5.2.4 Adapting the Grid and Order

This final table is the culmination of the paper and an example of WOFD2 with all options in use. The grid is adjusted between a maximum density of 128 and a minimum density of 64. The order of accuracy is adjusted between a maximum order of 16 and a minimum order of 8. The error converge in a nice manner toward the minimum error which occurs at the maximum grid density of 128 and the maximum order of accuracy of 16.

The usual Chebyshev grid is evenly-spaced in angle $\theta_i = i\pi/N$ for $i = 0, ..., N$. In the physical space the grid distribution is $x_i = \cos(\theta_i)$ which is shaped like a semi-circle. When one applies the wavelet grid adaptation to this evenly-spaced θ_i then obtains in the physical space the distribution $x_i = \cos(\theta_i)$ in the portion of the domain away from the pulse, and the twice-as-dense grid distribution $x_i = \cos(i\pi/(2N))$ in the portion of the domain near the pulse. Note that the grid is the usual Chebyshev grid near the boundary. It is only safely away from the boundary that the grid density

Grid Pts	t_f Grid	Order of Acc	L_2 Error	L_∞ Error	Thresh	Final Time
128/64	65	16	$2.95 * 10^{-4}$	$1.30 * 10^{-3}$	100.0	$\pi/2$
128/64	81	16	$1.57 * 10^{-3}$	$7.31 * 10^{-3}$	10^{-3}	$\pi/2$
128/64	87	16	$2.34 * 10^{-4}$	$8.16 * 10^{-4}$	10^{-4}	$\pi/2$
128/64	89	16	$2.43 * 10^{-5}$	$8.09 * 10^{-5}$	10^{-5}	$\pi/2$
128/64	87	16	$2.36 * 10^{-6}$	$9.20 * 10^{-6}$	10^{-6}	$\pi/2$
128/64	91	16	$3.18 * 10^{-7}$	$1.09 * 10^{-6}$	10^{-7}	$\pi/2$
128/64	93	16	$1.05 * 10^{-7}$	$4.66 * 10^{-7}$	10^{-8}	$\pi/2$
128	129	16	$4.26 * 10^{-8}$	$2.24 * 10^{-7}$	0.0	$\pi/2$

Table 8: WOFD2 Adaptive Grid, but Accuracy fixed at 16

Grid Density	t_f Grid	Order of Acc	L_2 Error	L_∞ Error	Thresh	Final Time
64	65	8	$3.82 * 10^{-3}$	$1.78 * 10^{-2}$	100.0	$\pi/2$
128/64	81	16/8	$1.61 * 10^{-3}$	$7.20 * 10^{-3}$	10^{-3}	$\pi/2$
128/64	80	16/8	$4.42 * 10^{-5}$	$2.68 * 10^{-4}$	10^{-4}	$\pi/2$
128/64	82	16/8	$6.28 * 10^{-6}$	$3.74 * 10^{-5}$	10^{-5}	$\pi/2$
128/64	84	16/8	$4.50 * 10^{-7}$	$2.64 * 10^{-6}$	10^{-6}	$\pi/2$
128/64	86	16/8	$1.23 * 10^{-7}$	$6.49 * 10^{-7}$	10^{-7}	$\pi/2$
128/64	87	16/8	$5.52 * 10^{-8}$	$2.25 * 10^{-7}$	10^{-8}	$\pi/2$
128/64	90	16/8	$5.12 * 10^{-8}$	$2.24 * 10^{-7}$	10^{-9}	$\pi/2$
128	129	16	$4.26 * 10^{-8}$	$2.24 * 10^{-7}$	0.0	$\pi/2$

Table 9: WOFD2 with Grid and Order Adaptation

makes an abrupt change in density. See Figure 2 for an example of an initial grid.

If the numerical scheme is working properly then the pulse will propagate to the middle of the domain and be similar in shape to the initial condition. The best measure of this similarity is the L_∞ error. At the final time the pulse will appear as in Figure 3.

The Chebyshev grid is naturally more dense near the boundaries than in the middle of the domain. With the wavelet adaptation of this Chebyshev grid, the grid points can be kept dense while maintaining a the Chebyshev distribution throughout most of the domain. Again, the most important region of the domain for a Chebyshev distribution is near the boundary. See Figure 4 for the grid distribution at the final time when the pulse has reached the middle of the domain.

Without grid refinement or order refinement the peak numerical error at the final time should be near the peak of the pulse, since it is this portion of the function which is most difficult to represent by polynomial interpolation. See Figure (5) for an example of such an error.

If the wavelet refinement threshold is not sufficiently low then one will see the peak error appear near a region of the domain where there is a grid or stencil discontinuity. A 'sufficiently low' refinement threshold will on the order of the L_∞ error when no grid or order refinement is executed. In Figure 6 noise is amplified at the interface where both the stencil and grid are refined. If the wavelet refinement threshold is adjusted to a smaller value then one can obtain an error profile similar to that in Figure (5).

When both the stencil and grid are changed throughout the calculation, one finds a relatively wide stencil near the peak value of the pulse. For the example of a 17 point stencil with accuracy of 16 at the pulse and a 9 point, accuracy 8, stencil away from the pulse see Figure (7).

6 Conclusion

This paper has covered many topics related to the construction of a very high order adaptive order and adaptive grid numerical method which has been named the Wavelet-Optimized Difference Method 2, or WOFD2. First it was necessary to explore the various ways in which difference operators can be constructed. This included a comparison of difference operators generated

from algebraic, trigonometric, exponential, and cosine polynomials. Next, which type of polynomial would be best for the construction of very high order numerical differencing. The conclusion, which is not a big surprise, is that one should use algebraic polynomials on Chebyshev grids. The next step was to apply wavelet grid and order adaptation in order to be able to reduce errors throughout the domain by either increasing the order of the numerical method or by increasing the grid density in the appropriate region. The results of the numerical tests were very positive and it appears that WOFD2 will applicable to a large range of numerical problems. The version of WOFD2 which has been presented here has been 'tweaked' very little. That is, it worked essentially for the first time it was tried. This is encouraging because most high order numerical methods require some kind of filtering or other refinement. Future plans for WOFD2 would be, perhaps, to try to find a proof of stability and to apply the method in higher dimensions.

References

[1] I. Babuska, "The p- and hp-Versions of the Finite Element Method: the State of the Art." *Finite Elements: Theory and Applications*, edited by D.L. Dwoyer, M.Y. Hussaini and R.G. Voigt. New York: Springer-Verlag. pp. 199-239 (1988).

[2] C. Canuto, M.Y. Hussaini, A. Quarteroni, T.A. Zang, (1988) "Spectral Methods in Fluid Dynamics", Springer-Verlag.

[3] M. Carpenter and D. Gottlieb, "Spectral Methods on Arbitrary Grids", ICASE Report No. 95-37, NASA CR-198158.

[4] S.N. Christofi, "The Study of Building Blocks for Essentually Non-Oscillatory (ENO) Schemes", PhD Thesis, Division of Applied Mathematics, Brown University, May 1996.

[5] G. Dahlquist and A. Bjorck, (1974) "Numerical Methods", Prentice-Hall.

[6] I. Daubechies, (1988) "Orthonormal Basis of Compactly Supported Wavelets", Comm. Pure Appl. Math., **41** pp. 909-996.

[7] P.J. Davis, (1975) "Interpolation and Approximation", Dover.

[8] W.S. Don and C.B. Quillen, "Numerical Simulation of Shock-Cylinder Interactions", Journal of Computational Physics, 122, 244-265 (1995).

[9] (1996) G. Erlebacher, M.Y. Hussaini, L. Jameson, "Wavelets: Theory and Applications", Oxford.

[10] D. Gottlieb and S.A. Orszag, (1977) "Numerical Analysis of Spectral Methods: Theory and Applications", SIAM-CBMS, Philadelphia.

[11] W. Gui and I. Babuska, (1986), "The h-, p- and hp-Versions of the Finite Element Method in One Dimension. Part I: The Error Analysis of the p-Version. Part II: The Error Analysis of the h and hp-Versions. Part III: The Adaptive hp-Version." *Numerische Mathematik*, Vol. 49, pp. 577-612; 613-657; 659-683.

[12] L. Jameson, "On The Wavelet Based Differentiation Matrix", Journal of Scientific Computing, September 1993 and ICASE Report No. 93-95, NASA CR-191583.

[13] L. Jameson, "On The Differentiation Matrix for Daubechies-Based Wavelets on an Interval", SIAM J. Sci. Comp., Vol. 17, Issue 2, 3/96 and ICASE Report No. 93-94, NASA CR-191582.

[14] L. Jameson, T.L. Jackson, D.G. Lasseigne, "Wavelets as a Numerical Tool", Proceedings from the Joint US-Japan Workshop on Combustion, 1993, J. Buckmaster, T. Takeno (eds.), Springer-Verlag.

[15] L. Jameson, "On the Wavelet-Optimized Finite Difference Method", ICASE Report No. 94-9, NASA CR-191601.

[16] L. Jameson, "Wavelet-based Grid Generation", 12-95, submitted to Journal of Computational Physics.

[17] G.E. Karniadakis and S.A. Orszag, Physics Today 34, (1993).

[18] H.O. Kreiss, technical report, University of Uppsala, Sweden, 1978.

[19] G. Strang and T. Nguyen, "Wavelets and Filter Banks", Wellesley-Cambridge Press, 1996.

[20] B. Szabo and I. Babuska, (1991) "Finite Element Analysis", Wiley Interscience.

[21] R.G. Voigt, D. Gottlieb, Y. Hussaini, (1984) "Spectral Methods for Partial Differential Equations", SIAM.

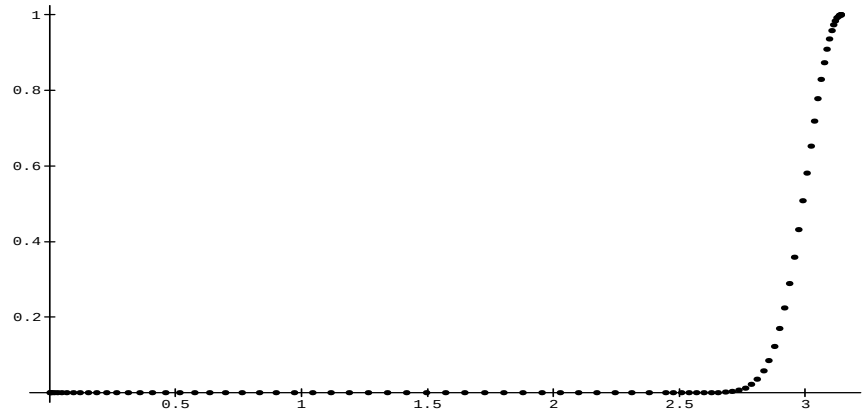

Figure 1: Initial Condition of Pulse Entering Domain

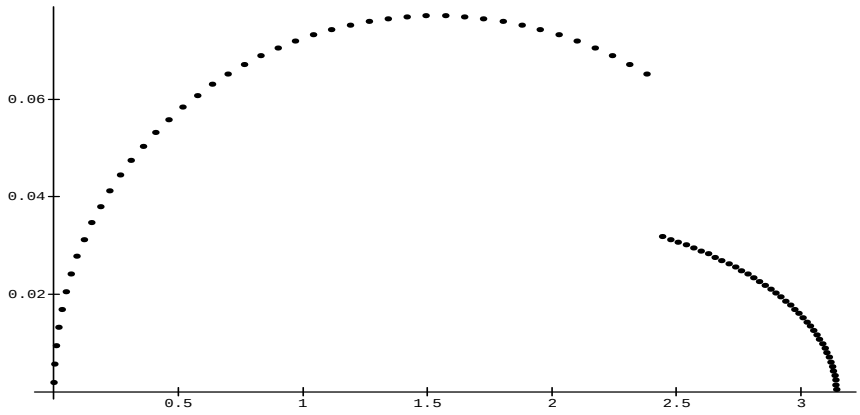

Figure 2: Initial Grid Density For Pulse Entering Domain

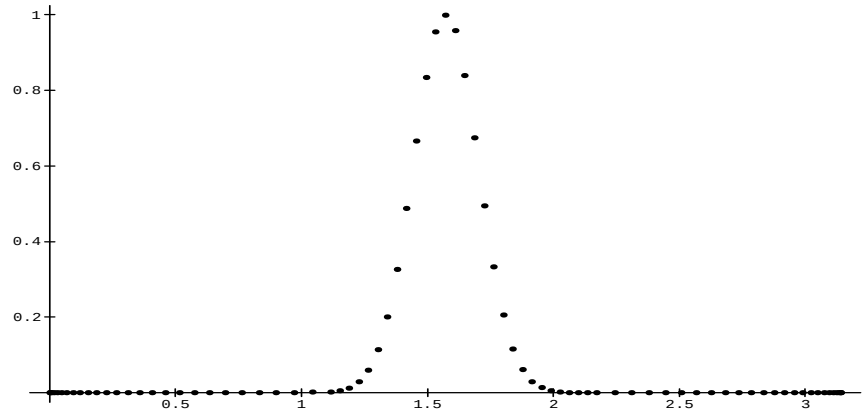

Figure 3: Pulse at Final Time

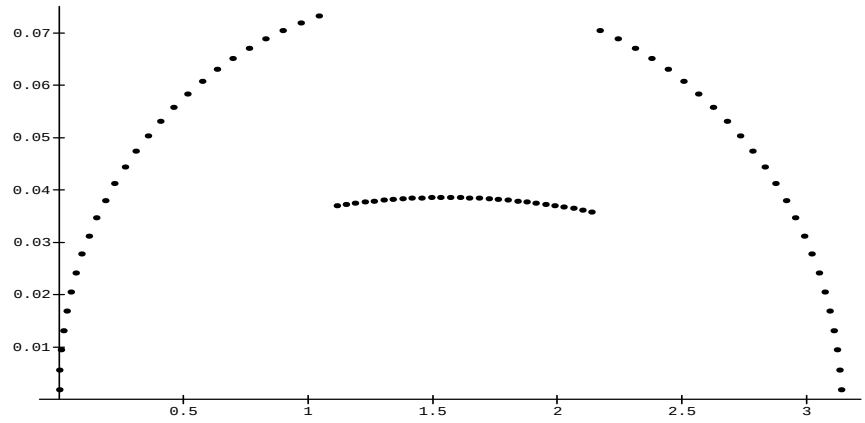

Figure 4: Adaptive Chebyshev Grid at Final Time

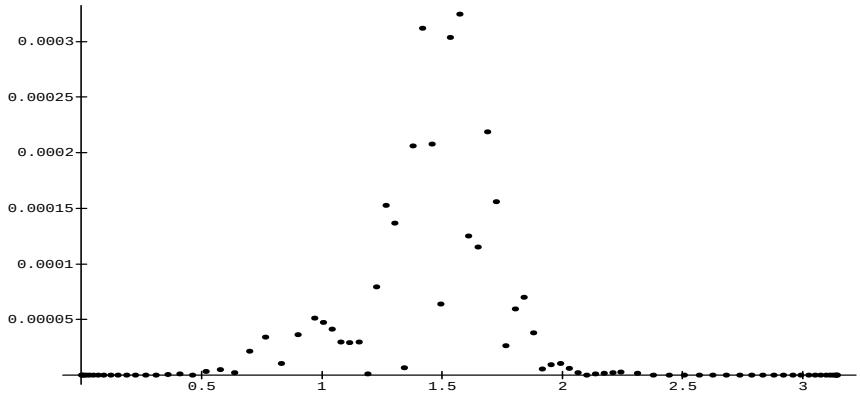

Figure 5: Typical Error at Pulse Peak

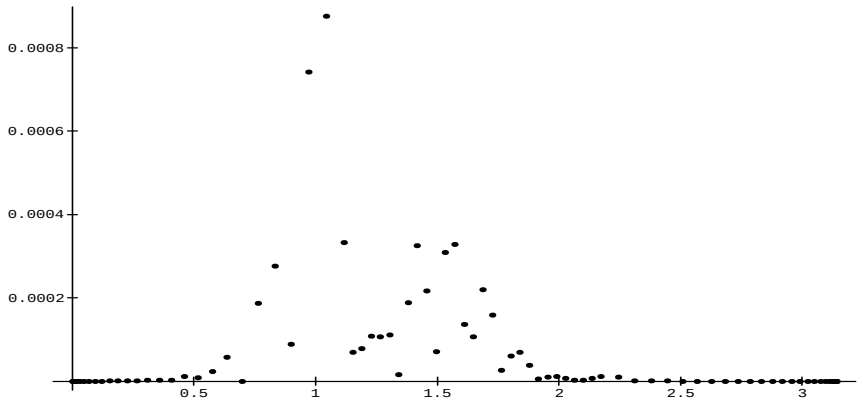

Figure 6: Typical Error at Stencil Discontinuity

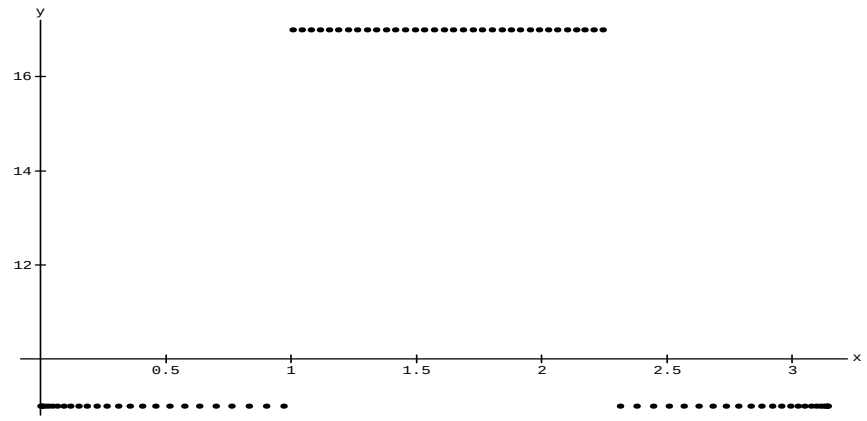

Figure 7: Width of Differencing Stencil at Final Time